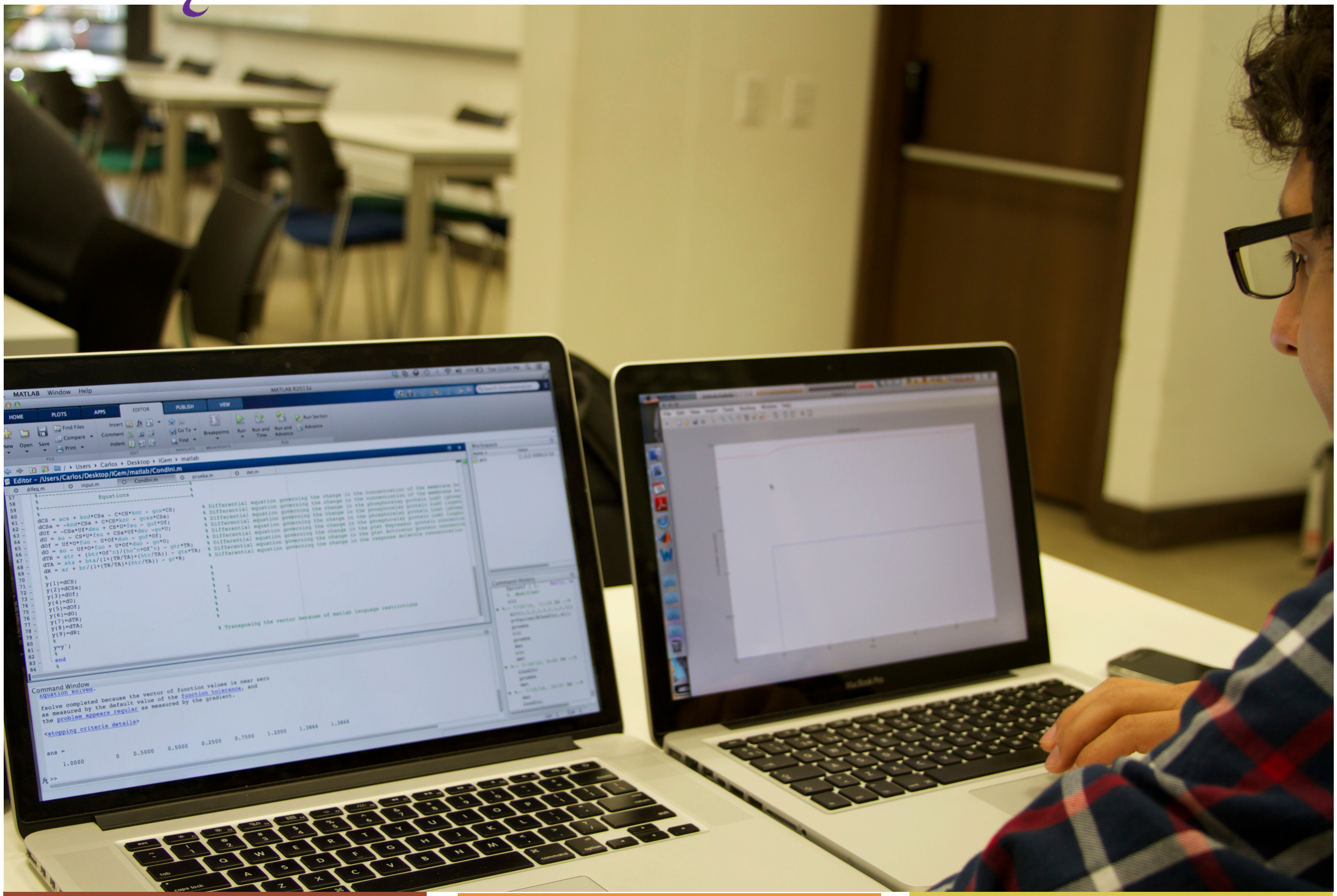
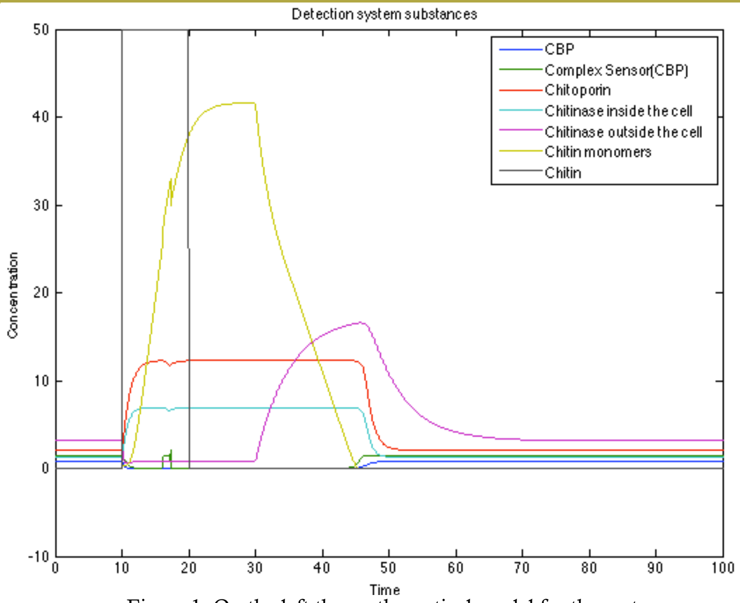
DO YOUR OWN MATHEMATICAL MODEL

Made by iGEM Colombia 2014



Most of the times when working in subjects around synthetic biology, one wishes to have a way to predict the dynamics of a certain system even before starting to work with it at the lab, maybe the system contains processes that at first sight seem complex and chaotic. Here are a few tips on how to model such systems:



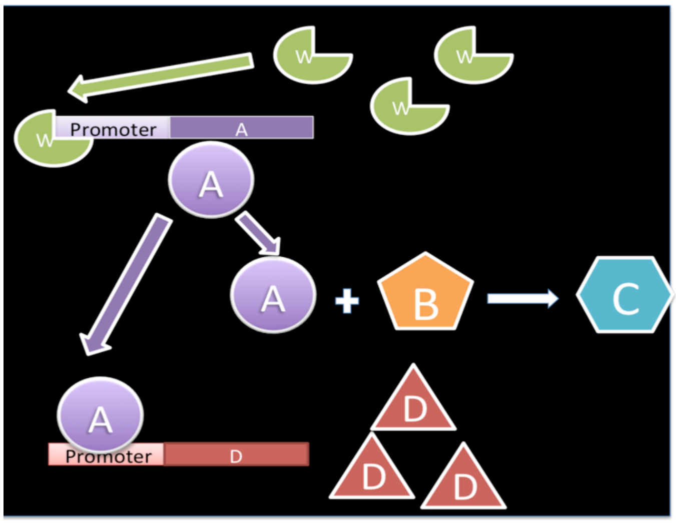


Figure 1. On the left the mathematical model for the system proposed by iGEM Colombia 2012. On the right system's example

The differential equations

The mathematical translation of biological systems

I. The first step in a mathematical model is to divide your system into simpler processes and define which parts take place and in which order. If it is possible define the process as a chemical reaction

EXAMPLE:
Look at the picture on the right; we have a system with **4 species** and we can describe **3** simple processes:

1. Production of A induced by W
2. A +B → C
3. Production of D induced by D

There are some processes that are no obvious in the system but they occur and need to be considered: **Production and destruction of proteins. Those always are represented by the reactions:**

*** → Protein (Production)**
Protein→ * (Degradation)

Continuing with our example we have to add the processes:

4. *→ W	8. A→ *
5. *→B	9. B → *
6. * → A	10. C → *
7. * → D	11 D→ *
	12 W→ *

When we are going to translate the processes into mathematical expression we use the law of mass conservation:

$$Acumulation = Input - Output + Production - Consmption$$

In a biological system, the accumulation stands for the concentration changes over time, the input and output are related with the processes of export and import of a molecule into the cell, the generation is related to the production by a gene or a chemical reaction, and the consumption is related to chemicals reactions too.

II. Once you have all your processes established it is time to write the law of mass conservation for **EACH SPECIE.** There must be a mathematical expression for all the processes involved.

EXAMPLE:

For the specie A of the example above we would have the following law of mass conservation:

Accumulation of A = Basal production + Production induced by W – Consumption in reaction – Degradation

There is no term for input or output and there are two processes involved in consumption

The accumulation always is expressed as a derivate in the form:

$$\frac{dA}{dt}$$

The following boxes show the mathematical expressions for the most common cellular processes:

Enzymatic processes

When a substance is produced (+) or consumed (-) in an enzymatic process we use the Michalis Menten equation:

$$r = \frac{Vmax[S]}{k + [S]}$$

Where [S] is the substrate of the enzyme, Vmax the maximum rate of reaction y k a number that defines the affinity Enzyme-Substrate

Basal production

Given by a rate constant!

$$r = \alpha$$

Natural degradation

It depends of the protein concentration and a rate constant:

$$r = \delta\,[A]$$

Chemical reactions

$$aA + bB \rightarrow cC + dD$$

Are expressed based on the law of mass action. Where the rate of the reaction is given by:

$$r = k\,[A]^a[B]^b$$

k is the kinetic constant.

REVERESE REACTIONS MUST BE CONSIDERED INDEPENDENT PROCESSES

Help: In the equation we always write the species that go on the tail of the reaction arrow. We use the stoichiometry coefficients as powers.

Gene activation

When a protein is produced due to the activation by a molecule we use a Hill type equation:

$$r = \frac{Vmax[S]^n}{k^n + [S]^n}$$

The terms are similar to Michaleis Menten´s. k represents the affinity of the DNA and the inductor and n is the cooperation number which stands for the number of union site to the DNA by the inducer

Gene Repression

Sometimes the production of a protein can be stopped by a repressor that block the promoter of the gene. For this we use another Hill type equation:

$$r = \frac{Vmax\ \ k^n}{k^n + [S]^n}$$

It is important to know that if we have a process of activation and repressionon the same protein we must add both hill type equations!!!

Transportation processes

There are some molecules that are transported from or to the cell. To represent these processes of input (+) and output (-) we use a first order kinetic expression:

$$r = \ m\,[A]$$

We write the concentration of A from where it is coming. If it is an import process [A] is the concentration outside the cell and if it is an export process [A] is the concentration inside the cell.

EXAMPLE:

If we write all the law of mass action for each species we would get the following **differential equations!!** ☺

$$\frac{dW}{dt} = \alpha_W - \delta_W[W]$$

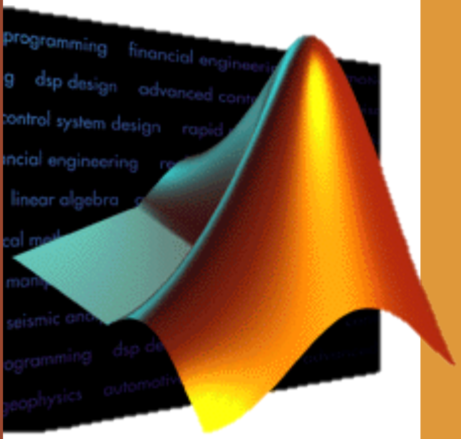
$$\frac{dA}{dt} = \alpha_A + \frac{Vmax_A[W]^{n_A}}{{k_A}^{n_A} + [W]^{n_A}} - k\,[A][B] - \delta_A[A]$$

$$\frac{dB}{dt} = \alpha_B - k\,[A][B] - \delta_B[B]$$

$$\frac{dC}{dt} = \ k\,[A][B] - \delta_C[C]$$

⚠ Be careful to always check the **units** of your expressions. They must be in: $\frac{Concentration}{Unit\ of\ time}$

SOLVING THE DIFFERENTIAL EQUATIONS



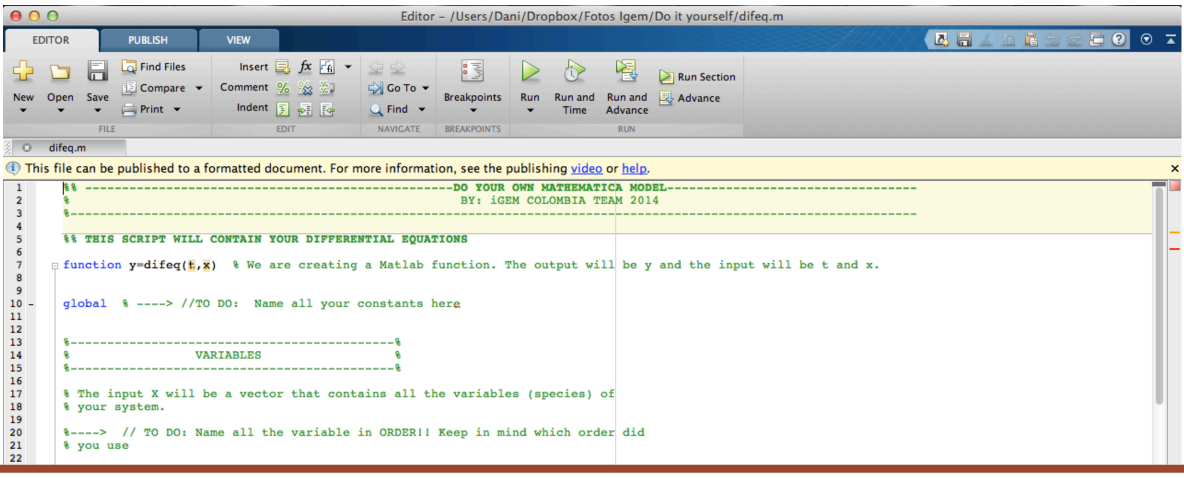
In this section we are going to explain step by step how to solve differential equation in Matlab.

We are going to use the scripts attached to this document. You will need to modify them as we move trough this section.

STAGE 1: WRITING THE DIFFERENTIAL EQUATIONS IN MATLAB

In the last section we made the differential equations for your system. In order to solve those equations we need to write a m file (Matlab file) with them.

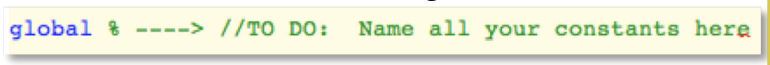
Open the file difeq.m…you will see this:



This file can be published to a formatted document. For more information, see the publishing video or help.

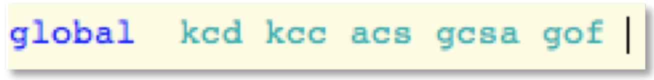
```
1    %% ---------------------------------------------------DO YOUR OWN MATHEMATICA MODEL-----------------------------------
2    %                                                      BY: iGEM COLOMBIA TEAM 2014
3    %--------------------------------------------------------------------------------------------------------------------
4
5    %% THIS SCRIPT WILL CONTAIN YOUR DIFFERENTIAL EQUATIONS
6
7    function y=difeq(t,x)   % We are creating a Matlab function. The output will be y and the input will be t and x.
8
9
10   global  % ----> //TO DO:  Name all your constants here
11
12
13   %--------------------------------------------%
14   %                 VARIABLES                  %
15   %--------------------------------------------%
16
17   % The input X will be a vector that contains all the variables (species) of
18   % your system.
19
20   %---->  // TO DO: Name all the variable in ORDER!! Keep in mind which order did
21   % you use
22
```

<u>II. Name all your parameters</u>: Go to line 10. You will see the word " global"



```
global % ----> //TO DO:  Name all your constants here
```

Write the name of your parameters and separate them by the space bar



```
global  kcd kcc acs gcsa gof |
```

Every time you see //TO DO: in the code you will have to erase the comment and fill the code according to the instructions.

III. Name all your variables: At line 7 the Matlab function starts: The output of the function will be y and the input will be t and x.

```
function y=difeq(t,x)
```

x is a vector that will contain all the variables of the system. In order to use them we need to name them. Go to Line 20 and fill the “TO DO:” like the example below:

```
% The input X will be
% your system.

Csa= x(1);
U= x(2);
Uf= x(3);
TR= x(4);
```

You can name them anyway you want. But you need to remember the order you use to do it.

If you have an input function you’ll have to declare the input molecules as a variable.

For this we call the input function as says in the explanation given at the last part of the section “Variable”.

```
C= input(t);
```

REMEMBER! When you call a function into another both must be in the same directory

INPUT FUNCTION

Some times we want our system to react only in the presence of certain molecule or input.

We can include this in the deterministic model if we express the concentration of the input molecule as a function of time. With it we will be able to see how the system behaves before, during and after the presence of the input.

The file input.m contains a script with a step function that returns the concentration of the input molecule depending of the time.

You can check the script and modify it. There are comments in it that will tell you how to do it.

IV. Write down your equations: Once the parameters and variables have been declared, it is time to write down all our equations.

First give a name to the equation, write equals (=) and then your equation using the same symbols and abbreviations you use in the last sections. ALWAYS finish the equation with “;”

```
%With the variables and parameters declare
%differential equations. We use the form:

% dA= your equation.
% dB = your equation
% BE CAREFUL WITH THE PARENTHESIS

dA = k*A - w*C;
dB= w*A*B+ vmax*S/(K+S);
```

a) Remember that Matlab can differentiate between upper and lower case letters.
b) BE CAREFUL WITH THE PARENTHESIS!!!

V. Write the output of the function: In order to make the equations readable for others m files we need to put them in the output vector y.

For this go to the section “Output” of the m file and write the out as shown in the Figure:

```
y(1)= dA;
y(2)= dB;
y(3)=dC;
```

STAGE 2: SOLVING THE DIFFERENTIAL EQUATIONS BY NUMERICAL APROXIMATION

Open the file det.m

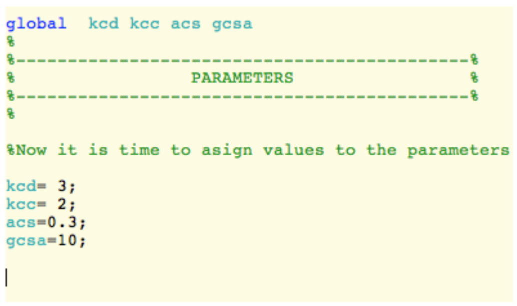
I. Assign value to the parameters:

Name all the parameters as you did on the step II of Stage 1.

The word global tells Matlab that we are going to use the same parameters on both files so we only have to assign their value once in all the process.

You have to assign the parameters the same name you did before.

Now go to Line 17 and for each parameter write the value (check section 3 for more information about how to get the values).



```
global  kcd kcc acs gcsa
%
%-----------------------------------------------%
%                   PARAMETERS                  %
%-----------------------------------------------%
%

%Now it is time to asign values to the parameters

kcd= 3;
kcc= 2;
acs=0.3;
gcsa=10;
```

Remember to end each line with ";"

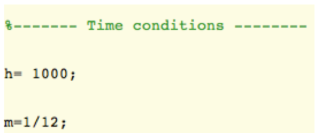
In order to know how your system behaves we developed a script that will solve the equations by a number approximation method called: 4th order Runge- Kutta.

With this script you will be able to change the simulation's maximum time and the initial conditions of the system.

Follow the steps and at the end you will have a figure like the one at the beginning of this guideline.

II. Establish the time conditions: When you run a simulation of a biological process you have establish the time range where you are going to evaluate the differential equations and the response of the system.

"h" represents the upper limit of the time range. Change this value as you want, but always keep in mind the units you are using.



```
%------- Time conditions --------

h= 1000;

m=1/12;
```

"m" represents the step length at which we move from the lower to the upper limit. The smaller the number the longer the time it will take the script to evaluate the equations.

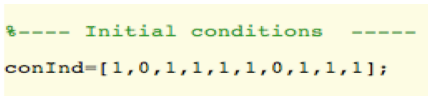
III. Initial conditions:

An important part of a numerical solution algorithm are the initial conditions. We can get different behaviors of our system if we use different initial conditions.

Usually we write low numbers like 0.1 or 1. But if you have a guess of how much the concentration of a certain protein should be inside the cell you should write that number.

In the script "condInd" is the vector that contains the initial conditions except for the input molecule.

It must have the initial values for all your variables in the **SAMER ORDER** that used in the ***det.m*** file.



```
%---- Initial conditions  -----

conInd=[1,0,1,1,1,1,0,1,1,1];
```

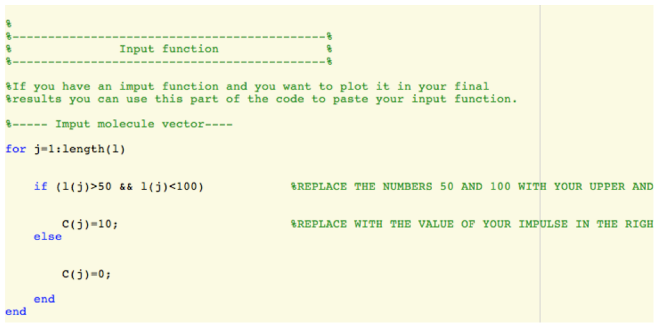
Note: Some times when running the script you won't get the desired response, changing the initial conditions will help you.

THE HARDEST PART IS OVER!!!

Now we will see how to modify the script in order to extract the information after the Runge_Kutta is done.

If you have an input function and you want to plot it with your response you'll have to declare the input molecules as a variable in the script.

Follow the instructions that are on the script at the Input Function section

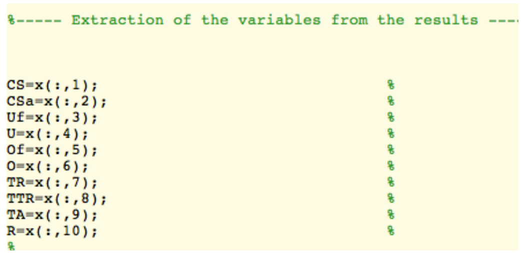


```
%
%---------------------------------------------%
%               Input function                %
%---------------------------------------------%

%If you have an imput function and you want to plot it in your final
%results you can use this part of the code to paste your input function.

%----- Imput molecule vector----

for j=1:length(l)


    if (l(j)>50 && l(j)<100)          %REPLACE THE NUMBERS 50 AND 100 WITH YOUR UPPER AND


        C(j)=10;                      %REPLACE WITH THE VALUE OF YOUR IMPULSE IN THE RIGH
    else


        C(j)=0;

    end
end
```

IV Plotting the results:

After running the Runge-Kutta algorithm we will get an *x* matrix where each column contains the concentration over time of each of your variables.

First we need to split the matrix into vectors for each of parameters, following the instructions that are on the script. At the end your code should look something like this:



```
%----- Extraction of the variables from the results ---



CS=x(:,1);                                  %
CSa=x(:,2);                                 %
Uf=x(:,3);                                  %
U=x(:,4);                                   %
Of=x(:,5);                                  %
O=x(:,6);                                   %
TR=x(:,7);                                  %
TTR=x(:,8);                                 %
TA=x(:,9);                                  %
R=x(:,10);                                  %
%
```

After this we only have to plot the results!

HOW TO PLOT THE RESULTS?

- The function "plot ()" will help us with this.
- Inside the parenthesis we write the name of the two vectors we want to plot. First the one that goes on the *x*-axes and then the one that goes on the *y-axes*.

```
plot(x,y)
```

- If you want to plot more vector in the same Figure you can add them by pairs of vectors. Always writing first the one that goes on the x-axis.

```
plot(x,y,x,z,x,w)
```

- Remember that in our script the time vector is called "l".

We are ready to run our script!!

On the toolbar of the editor window there is a play button.

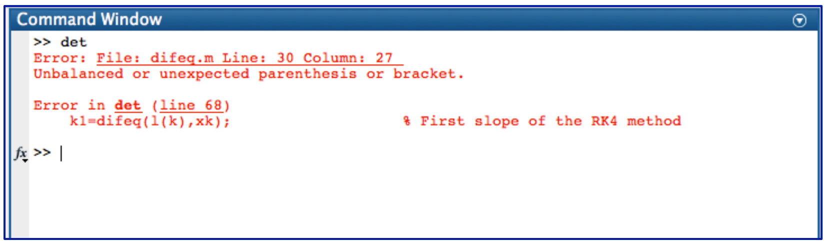
CLICK IT!

Some tips!

- Your script probably won't run properly the first time. But do not worry probably you have some lame mistakes.

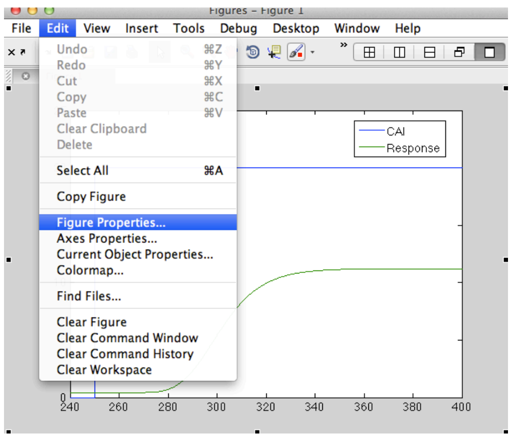
  If that happens to you, go to the main window of Matlab and read the red message that appears on the "Command Window"; it will tell you the problem.

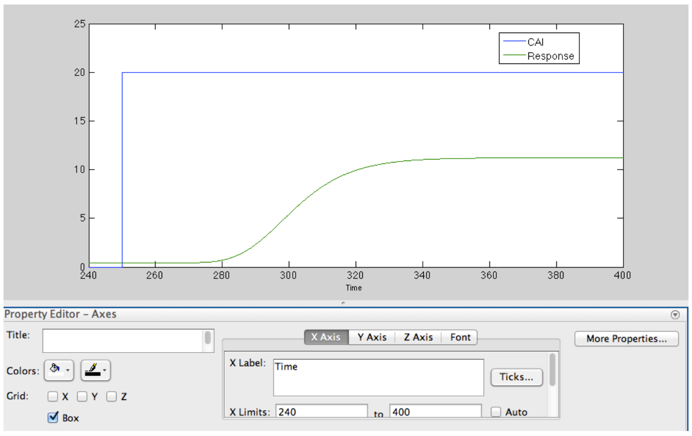
  For example the picture below shows us a common mistake. We have some open parenthesis at Line 30 of the difeq.m scrip that we forgot to close.



```
Command Window
>> det
Error: File: difeq.m Line: 30 Column: 27
Unbalanced or unexpected parenthesis or bracket.

Error in det (line 68)
    k1=difeq(l(k),xk);                  % First slope of the RK4 method

fx >> |
```

- If you want, you can add legends and titles to your Figure, once is created. Go to the menu bar→ Edit → Figure Properties.

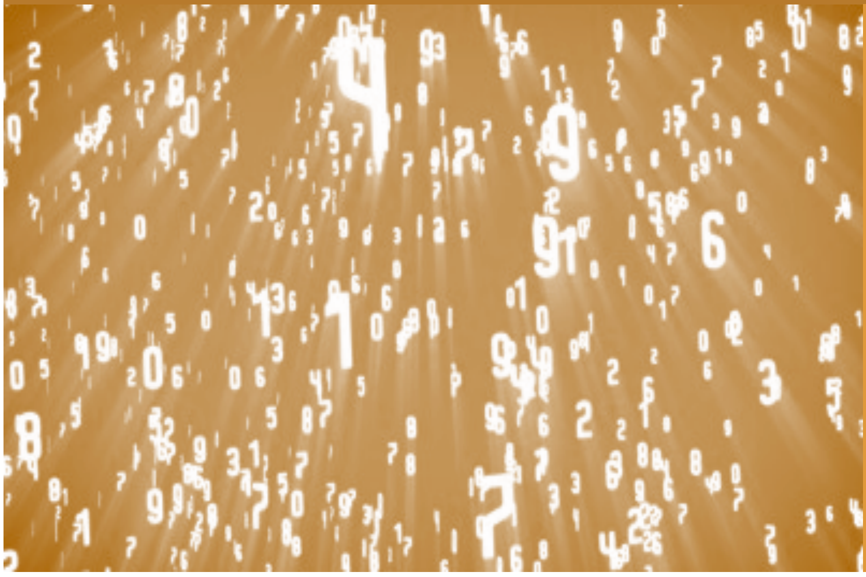




Property Editor – Axes
Title:
Colors:
Grid: ☐ X ☐ Y ☐ Z
☑ Box
X Label: Time
X Limits: 240 to 400 ☐ Auto

Click on the Figure's part you want to modify and a menu like the picture on the right will appear. Make it as pretty as you want ☺!

THE PARAMETERS



As we saw in section one, when we made differential equations, there are a lot of parameters that define the system.

If the real values for the constants are unknown, the system cannot have any biological sense.

In this section we will make an approximation to the right set of parameters

THERE ARE THREE POSSIBLE WAYS TO FIND THE PARAMETERS

Guessing

This should be your last resource, but most of the time is the only thing we can do.

Here are some advices so you can guess your parameter properly:

1. **Estimate an order of magnitude based on the literature:** Maybe you wont find the parameter for your system but there may be another similar parameter for a different system.

2. **Perform a sensitivity analysis:** This kind of analysis allows us to see how sensitive is the model to a parameter.

   If the response is no sensitive to the parameter you can leave it with the value you chose at first.

   But if the response changes a lot when the parameter changes you should try to measure it at the lab or at least fence it in a small range.

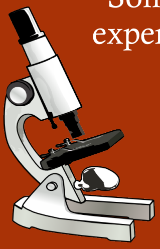
   The next page shows you how to preform the Sensitive analysis in Matlab.

Literature

There are a lot of studies trying to find biological parameters, such as basal rate of protein production, kinetic constants, Monod's constants, etc.

Try to find as much parameters as you can in the literature. This may be boring and time consuming but it is the best way to get a good model

Experiments



Some times it is possible to design experiments at the lab that helps us to get some of the unknown parameters.

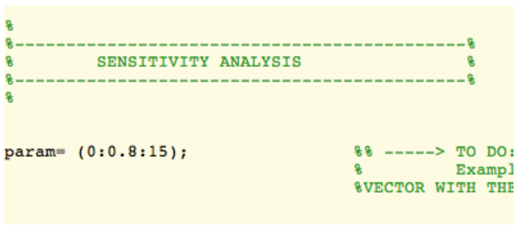
If you can do this, do not hesitate!!

SENSITIVITY ANALYSIS

Open the file Sensitivity_Analisis.m

THIS FILE IS A MODIFICATION OF THE *det.m*, HERE WE SHOW THE NEW CHANGES YOU HAVE TO MAKE.

I. First you will have to estimate the order of magnitude and an appropriate range to vary the value of the parameter. Usually we vary the guessed value around two orders of magnitude.

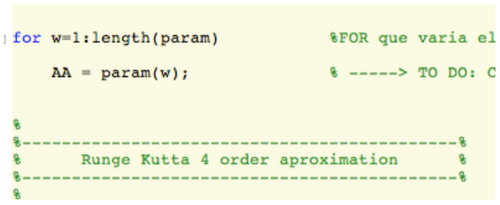


```
%
%-----------------------------------------------%
%          SENSITIVITY ANALYSIS                 %
%-----------------------------------------------%
%


param= (0:0.8:15);                    %% -----> TO DO:
                                      %         Exampl
                                      %VECTOR WITH THE
```

In the section of "Sensitivity analysis", the variable "param=" contains the vector with the values of the parameters at which we are going to evaluate the model.

II. Continue modifying the script as you did on the *det.m* file.

After the first for you will have to change the letter AA, for the name of the parameter you are varying.



```
for w=1:length(param)                 %FOR que varia el

    AA = param(w);                    % -----> TO DO: C


%
%-----------------------------------------------%
%       Runge Kutta 4 order aproximation        %
%-----------------------------------------------%
%
```

THIS TIME YOU DO NOT HAVE TO PLOT ANYTHING!!!

The code will do it for you. The x-axes will be the different values of the parameter and the y-axis the response you chose on step III

III. This time on the section "Extraction of the variables" after splitting the *x* matrix you will have to save a result on the *rta* matrix.

Which result are you going to save depend on your system.

Usually we save the concentration of a variable at a time of interest or the difference of a variable's concentration before and after the input.

Check the script and follow the instructions to save your response

THE END!!
We hope this helps you to perform your mathematical model

As you all know nature is so magnificent and complex that every day we discover new thing about her.

The tutorials youn find in here, only take into considerations the more common mathematical models used in iGEM projects during the past years.

If you need something more complex or something different we either invite you to read some of the literature like:

Alon, U. (2006). *An introduction to systems biology: design principles of biological circuits*. CRC press.

IGEM COLOMBIA 2014
UNIVERSIDAD DE LOS ANDES
BOGOTÁ, COLOMBIA

If you need more information feel free to contact:

Dr. Juan Manuel Pedraza Leal
jmp@uniandes.edu.co

Daniela Olivera Mesa
d.olivera1320@uniandes.edu.co

Roberto Morán Tovar
r.moran1292@uniandes.edu.co